LAB 5

STAT 28

February 16, 2016

Welcome to LAB 5! In this lab, we will:

- Learn about how to calculate confidence intervals (CIs) for t-tests.
- Learn how to create confidence interval using bootstrap methods.
- Learn basic syntax of linear regression.
- Review how to do bootstrap on the coefficients of linear regression.

In this lab, we will look at a comic book data created by FiveThirtyEight for their story *Comic Books Are Still Made By Men, For Men And About Men*, where they claimed that Comic books vastly under-represent women. Specifically, they said that woman characters have lower appearance counts after analyzing the Marvel and DC dataset. Well, is that true? Let's apply our testing methods to their analysis!

```r
# Read in data.
marvel <- read.csv("marvel-wikia-data.csv")
```

The data `marvel-wikia-data.csv` comes from Marvel Wikia. It has the following variables:

- `page_id`: The unique identifier for that characters page within the wikia
- `name`: The name of the character
- `urlslug`: The unique url within the wikia that takes you to the character
- `ID`: The identity status of the character (Secret Identity, Public identity, [on marvel only: No Dual Identity])
- `ALIGN`: If the character is Good, Bad or Neutral
- `EYE`: Eye color of the character
- `HAIR`: Hair color of the character
- `SEX`: Sex of the character (e.g. Male, Female, etc.)
- `GSM`: If the character is a gender or sexual minority (e.g. Homosexual characters, bisexual characters)
- `ALIVE`: If the character is alive or deceased
- `APPEARANCES`: The number of appareances of the character in comic books (as of Sep. 2, 2014. Number will become increasingly out of date as time goes on.)
- `FIRST APPEARANCE`: The month and year of the character's first appearance in a comic book, if available
- `YEAR`: The year of the character's first appearance in a comic book, if available

We will focus on two columns `SEX` and `APPEARANCES`. To make thing simpler, we created two vectors `female.logappearances` and `male.logappearances` for you, which contains the log appearances counts for female and male characters separately. You will use these two vectors to do hypothesis testing in the rest of the lab.

```r
female.logappearances <- log(marvel$APPEARANCES[marvel$SEX == "Female Characters"])
male.logappearances <- log(marvel$APPEARANCES[marvel$SEX == "Male Characters"])
```

T-test confidence intervals

Exercise 1.

(a) Get the confidence interval for the mean of the female and male log appearance using t.test function.

```r
# Insert your code here for calculating the CI, and save the CIs as
# `log.female.ci` and `log.male.ci`
# log.female.ci <-
# log.male.ci <-
```

You may found that when you print the confidence interval, there is an additional line named `attr(,"conf.level")` with value 0.95. In R, all objects can have arbitrary additional attributes, used to store metadata about the object. Attributes can be accessed using `attributes()` or `attr()`. For example, to get all the attributes of `log.male.ci`: `attributes(log.male.ci)`. And to get the "conf.level" attribute of `log.male.ci`: `attr(log.male.ci, "conf.level")`. Attributes do not influence the fact that `log.male.ci` is a two-dimensional vector.

(b) Get the confidence interval for the difference in the log appearance count for female and male using t.test function.

```r
# Insert your code here for calculating the CI, and save the CI as
# `log.diff.ci`
# log.diff.ci <-
```

(c) Based on your calculation, you prefer which of the following statement: A. There is no significant difference in appearance counts for male and female in Marvel comics. B. The female character appearances count is significantly larger than the male in Marvel comics. C. The male character appearances count is significantly larger than the female in Marvel comics.

```r
# Uncomment the line of your answer for this question:
# Ex1c.answer <- "A"
# Ex1c.answer <- "B"
# Ex1c.answer <- "C"
```

Bootstrap confidence intervals

Exercise 2.

In this exercise, you will use bootstrap to get the confidence interval of the t-statistic between female and male log appearance count.

(a) Caculate the observed t statistics.

```r
# Insert your code here for calculating the observed t statistic
# `obs.t`
# obs.t <-
```

(b) Complete the following function to calculate the bootstrapped t-statistic. The function only needs to calculate one bootstrapped statistic, and you will replicate it later.

```r
# Complete the function for calculating the bootstraped the t-statistic
bootOnce <- function() {
  # boot.male <-
  # boot.female <-
  # t.stat <-
  # return(t.stat)
}
```

You can test whether your code work or not by run the following chunk:

```r
# test <- bootOnce()
# test  # This should give you a numeric value, which is a bootstrapped statstic
```

(c) Replicate the function `bootOnce` 1000 times to get 1000 bootstrapped statistics.

```r
# Insert your code here and save the bootstrapped t statistic as
# `boot.t`
# boot.t <-
```

(d) Plot the histogram of your bootstrapped statistics.

```r
# Insert your code for histogram here
```

(e) Calculate the bootstrapped CI under the 0.95 confidence level. HINT: use `quantile` function to find the bounds. lower bound = (1-0.95)/2 quantile. upper bound = 1 - (1-0.95)/2 quantile.

```r
# Insert your code and save your bootstrapped CI as
# `boot.ci`
# boot.ci <-
```

(f) Based on your calculation, you prefer which of the following statement: A. I will accept the hypothesis that there is no significant difference in appearance counts for male and female. B. I will reject the hypothesis that there is no significant difference in appearance counts for male and female.

```r
# Uncomment the line of your answer for this question:
# Ex2f.answer <- "A"
# Ex2f.answer <- "B"
```

Linear regression

Let us first look at a simulated example. You can ignore the following process about how the data was generated

```r
# Make some data
# X increases (noisily)
# Y is constructed so it is inversely related to xvar
set.seed(955)
xvar <- 1:20 + rnorm(20,sd=3)
yvar <- -2 * xvar + 3 + rnorm(20,sd=4)

# Make a data frame with the variables
dat <- data.frame(x=xvar, y=yvar)
# Show first few rows
head(dat)
```

```
##           x          y
## 1 -4.252354  14.802251
## 2  1.702318  -3.063383
## 3  4.323054  -9.771777
## 4  1.780628  -3.130846
## 5 11.537348 -25.120776
## 6  6.672130 -13.533446
```

To do regression on y using x as a predictor, we can call the `lm` function:

```r
# These two commands is equivalent
fit <- lm(y ~ x, data=dat)
fit <- lm(dat$y ~ dat$x)
```

To get the detailed information about the fit, such as coefficient estimates, t-statistics and p-values.

```r
summary(fit)
```

```
##
## Call:
## lm(formula = dat$y ~ dat$x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0503 -2.6031 -0.2142  2.6217  7.6811
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.7791     1.4737   1.886   0.0756 .
## dat$x        -1.9889     0.1295 -15.355 8.71e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.642 on 18 degrees of freedom
## Multiple R-squared:  0.9291, Adjusted R-squared:  0.9251
## F-statistic: 235.8 on 1 and 18 DF,  p-value: 8.709e-12
```

The coefficients estimation can be accessed by:

```r
fit$coefficients
```

```
## (Intercept)       dat$x
##    2.779061   -1.988899
```

To get the estimated coefficent of the intercept term:

```r
fit$coefficients[1]
```

```
## (Intercept)
##    2.779061
```

To get the estimated coefficent of the slope term:

```r
fit$coefficients[2]
```

```
##     dat$x
## -1.988899
```

Google Trends is a web facility provided by Google, based on the volume of Google search. It provide equally spaced time series data of the search volume You may compare different topics to discover how peoples' interests change over time. You've learned two most widely-used programming languages (R and Python) for data science. But which language is more popular over time?

The following picture shows how to retrieve the Google Trends data. You may download and analysis your topics of interests as well. The dataset we obtained contains the following variables:

- **week**: beginning date of the week (recent 5 years)
- **python**: trend of the search term **Data science Python**
- **r**: trend of the search term **Data science r**

Read the data.

```r
data_science <- read.csv("data_science.csv")
# convert string to date object
data_science$week <- as.Date(data_science$week, "%Y-%m-%d")
# create a numeric column representing the time
```

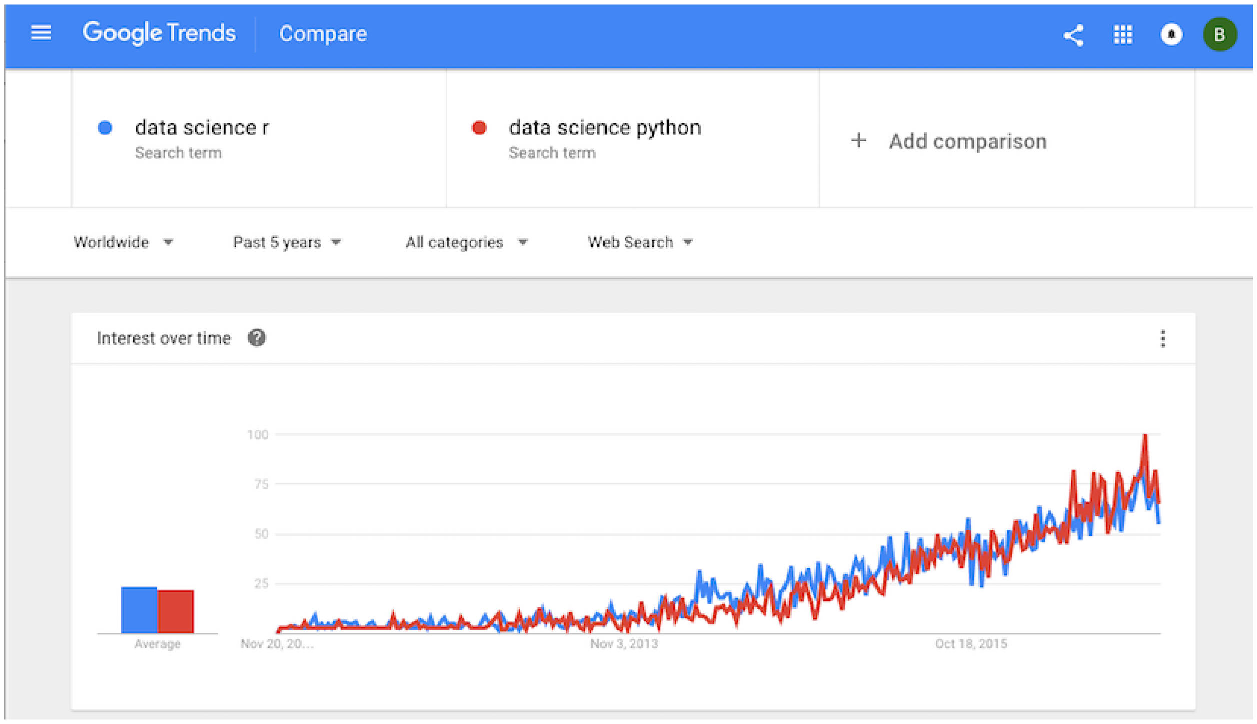

Figure 1:

```
data_science$time <- as.numeric(data_science$week)
data_science$time <- data_science$time - data_science$time[1] + 1
```

Exercise 3

Do regression on the r index using the predictor `time`. Get the estimate of the slope.

```
# Insert your code here and save the estimated slope as
# `r.slope`
# r.slope <-
# r.slope
```

Exercise Bonus

This exercise is not required. But you will find it helpful for your HW3. It is a review on how to get the bootstrapped CI for slope term in regression.

(a) Complete the following function to calculate the bootstrapped slope estimate. The function only needs to calculate one bootstrapped estimate, and you will replicate it later.

```
# Complete the function for calculating the bootstraped slope estimate
bootOnce <- function() {
  # # sample row indices with replacement from the data_science data frame
  # # HINT1: row indices: 1:nrow(data_science)
  # # HINT2: you will need nrow(data_science) samples for the bootstrap
  # boot.sample.indices <-
  # # use `boot.sample.indices` to select corresponding rows from `data_science`
  # boot.sample <-
  # # calculate bootstrapped slop using data frame `boot.sample`
  # boot.stat <-
```

```r
  # return(boot.stat)
}
```

(b) Replicate the function `bootOnce` 1000 times to get 1000 bootstrapped slope.

```r
# Insert your code here and save the bootstrapped t statistic as
# `boot.slope`
# boot.slope <-
```

(c) Calculate the bootstrapped CI under the 0.95 confidence level.

```r
# Insert your code and save your bootstrapped CI as
# `boot.slope.ci`
# boot.slope.ci <-
# boot.slope.ci
```